Starting Computing Final Project Report

Collin Sinclair

July 18, 2020

Challenges

There were three main areas that required prolonged problem solving and deep thought. Two were related to file I/O. The first was a more general problem: I knew that I wanted to keep the user data and tweets in an external file so that the information could be saved across sessions, but deciding how I wanted to implement this (or how I *could* implement this) brought with it many brainstorming sessions, trials and errors, and some hand-drawn diagrams to help me understand the algorithm I eventually settled on. Then, after much copying and pasting, I decided to write a function to read files and a function to write them. This way, the process was streamlined and consistent.

The second roadblock I encountered was a bit more specific. One of my goals was to allow a user to follow others then save whoever they were following to the user data text file. This would allow users to have a non-volatile following list, and they wouldn't be required to re-follow others every time they logged on. Initially my approach seemed sound, but something was failing as the following list wasn't being saved. After many hours of inspection, I determined the root of my problem to be that the driver program and the class program were referring to two different data structures, and thus the information was not passing from memory to storage correctly. Once I identified the source of the problem, it was easy enough to fix.

Finally, I had a tricky time writing the search function for the *Explore* feature, which shows a user all of the tweets written by non-followees. Similarly to the second problem, the issue had to do with what variable I was editing; it was the difference between operating on an iterator and actual item in a list. I'm always tempted to use the `auto` type, but it turns out that that's not always the best option!

Insights

Despite my familiarity with the object-oriented language Python, I learned a *lot* about classes and objects throughout this course, and specifically while working on this project. It became clear to me how powerful this kind of programming could be, and I grew less and less intimidated the complexity involved in OOP. I also felt a sort of freedom working with classes, in the sense that I could write modular pieces of code and then call from those pieces in the main program and know that they would work the way I expected them to. This process made the project seem much more manageable in that I could work on small bits at a time and then tie everything together. Additionally, though a C++-specific skill, I gained much comfort in working with header files, something that I didn't quite grasp the first time we learned about them in class.

Another area into which I gained insight was the design of a very, *very* basic user interface. I wanted to write the program in such a way that anyone, namely non-programmers, could sit down and enjoy a few minutes using "Twitter". Thus, I had to create an exhaustive menu system with plenty of input validation so that the user could easily navigate the program without fear of screwing something up. In the process of putting together this "UI" (if you can even call it that), I learned a lot about the importance of organizing my code in a way that was understandable and accessible. In other words, designing a user interface also forced my to uphold good programming practices, disallowing me to be lazy or messy.

If I Had More Time

Given more time to work, there are several features (perhaps too many to list here) I would like to add to my Twitter program. First, the program in its current state allows the user to follow others and view the number of people they have followed. Eventually, I would like to implement a way for users to view the number of people who follow them (and perhaps a followers list, as well). At the moment, I haven't thought of the best way to do this – I suppose I could always use a search, but maybe there is a better way.

On top of a followers list, I would love to allow

users to direct message (DM) one another. Now, for a program that only runs locally and in one instance at a time, this may not be super useful. It might be a fun way to send secret messages to other users though, without posting it as a Tweet so that everyone could see it.

Additionally, given the program somehow got large enough and had sufficient content to warrant this, I would implement a more user-specific explore feature that considers the user's interests (provided at account creation, I suppose). I would likely achieve this through the use of a keyword search.

While creating and using the Twitter program, I always thought it was funny that the passwords were stored in plain text in the same directory as the program. So while it looks and feels official in at the command line, the whole idea of a password is kind of useless security-wise the way I've written it. I think it would be fun to somehow encrypt the passwords (and probably all of the user data while I'm at it) so that tech-savvy individuals couldn't simply view the passwords of all the views who have created accounts.

Finally, the ultimate culmination of my project would be to connect it to the internet, such that users across the globe could use it in a similar way to the fashion in which real Twitter is used. Of course, I wouldn't expect it to gain any traction outside of the people who are kind enough to beta-test the program for me, but it would still be very cool to add network connectivity. In doing this, I should probably also refresh the users and tweets feeds regularly, so that if person A posts in location A, then person B in location B can see it without having to restart the program.

Resources

The Project Checkpoint meeting and Resources lecture were indubitably helpful in guiding my in the development of my project. At the checkpoint, Srinjita gave me many useful pointers about ways to store and access the data in text files for program use. Unfortunately I was unable to implement all of these, but they are on my radar for the next time I start a large scale project.

In the Project Resources lecture we learned about the `switch` control structure, which ended up being the primary avenue by which the user interacted with the program. As such, my program would not exist the way it does now without the information from that lecture. I also make use of `for-range` loops quite a bit (which we reviewed in that lecture), and this also opened the door to use of the `auto` type.

In all, I felt very supported during the development of my final project, and I am grateful for all of the resources that were available to us as students that allowed us to make our projects even better.

References

Other than slides and examples from class, I used documentation from The C++ Resources Network quite a bit. Though not a resource when writing the actual code, the development of my project would not have been possible without the help of select friends and family members who beta-tested my work and helped uncover problems that needed to be fixed, as well as features that should be implemented.